DOCUMENT RESUME

ED 144 615 IR 005 263

AUTHOR Lofstrom, Mats; And Others
TITLE 3RIP: Data Structures for Text Files.
INSTITUTION Royal Inst. of Tech., Stockholm (Sweden). Library.
REPORT NO TRITA-LIB-4061
PUB DATE Jan 77
NOTE 18p.; For related documents, see IR 005 260-261

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS Computers; Data; *Information Retrieval; *Information Science; Information Storage; *Information Systems
IDENTIFIERS 3RIP Search System; BLISS Programing Language; DEC 10 Computer

ABSTRACT

A record format and file structure for searching and editing of large textual files have been designed within the 3RIP project. This document describes: (1) general characteristics of the system; (2) implementation of the system on a DEC 10 computer in the programming language, BLISS; (3) external record format; (4) internal format; and (5) file structure and file maintenance. (Author/AP)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED144615

REPORT TRITA-LIB-4061 JUNE 1977

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

3RIP: DATA STRUCTURES FOR TEXT FILES

Mats Löfström, Jan Sunnebäck, Christer Bryntesson,
Bo Johansson, Rolf Larsson

Royal Institute of Technology Library
S-100 44 Stockholm 70

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Stephan Schwarz

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

IR005 263

3RIP: DATA STRUCTURES FOR TEXT FILES

Mats Löfström, Jan Sunnebäck, Christer Bryntesson,
Bo Johansson, Rolf Larsson

SUMMARY

A record format and file structure for searching and editing of large textual files have been designed within the 3RIP project.

A record is structured into one or more data fields, each divided into one or more subfields. A subfield is a character string of variable size.

Two equivalent structures are used to represent a record. For processing in primary storage an internal format is used, where each data field is represented by a list structure. The external format provides a compact representation used when records are stored in files on secondary storage.

Except the records proper each file consists of a directory, through which records are accessed, and a control record, where parameters needed for the processing of the file are stored. A record is named by a unique alphanumeric string, and the directory is arranged as a B-tree structure ordered on the name strings.

A 3RIP file of some four million records with an average size of 500 characters can be stored using $0.9\ 10^9$ 7-bit bytes. Two accesses are needed to reach a record in a file of some 1.6 million records.

This report describes the file structure and record format for text files used within the 3RIP system. In section I the objectives governing the design and the general characteristics of the file and record structures are presented. Section II gives a detailed description of how the structures are implemented on a DEC-10 computer.

I. GENERAL CHARACTERISTICS

The 3RIP system (1) is an interactive retrieval system for large textual files, and 3RIP also contains a conventional text editing system used to create and modify records within system files. As the files are large a compact format for their external storage is desirable. As editing is to be allowed on all records the file structure must permit rapid access to each individual record in order to obtain efficient updating. Moreover the records must be structured so that the contents of a record can be modified with little difficulty. Several files are handled by the 3RIP editor, e g the primary file, the alphabetically ordered keyword file, private text files, and files used to help the user to run the system. The 3RIP system is intended for a number of different applications where large textual files have to be searched and processed. This means that the record format must permit a great variety of information structures to be represented within a record.

To meet these requirements a special record format and file structure, in the following referred to as the 3RIP record format and the 3RIP file structure, have been designed. With the exception of the inverted file, described in (2), and some files used for system

control, all files within the 3RIP system conform to this structure.

All 3RIP records consist of two parts:
- a data part, divided into one or more data fields, each divided into one or more subfields.
- a control part, containing a record identifier and for each data field indicating locations and sizes of its subfields.

A subfield is an ASCII-coded character string of 7-bit bytes. A subfield may be of any size and a record may contain any number of data fields and subfields.

Within each record one data field is reserved and may be used to indicate if any of the other data fields of the record are to be treated in some special way when processed. Thus more complex data structures can be represented within a record.

Two equivalent structures, an external form, EFORM, and an internal form, IFORM, are used to represent 3RIP records. The objectives governing the design of the structures have been for EFORM to minimize the space requirements of a record when stored on secondary storage and for IFORM to facilitate processing in primary storage of the individual data fields and subfields of a record. Characteristic of the EFORM is that the control part is relatively small and that the data part is stored in one string. The last feature is of importance as it facilitates the use of the compression method of 3RIP (3), by which the space required to store the data part is about halved.

When a record is to be processed, the EFORM representation of the record is read into primary storage, where the data part is stored into a buffer. Then the control part of the record is transformed from the EFORM to the IFORM and if the data part is compressed it is decoded also.

Within IFORM each data field is represented by a doubly-linked list (4), the list head of which is pointing to a chain of nodes, each containing the location of the first character of a subfield and its length in bytes.

Through these list structures efficient procedures for IFORM editing have been implemented. When a new subfield is inserted the character string constituting the subfield is appended to the buffer and a new node is linked into its proper position within the corresponding list. When character strings are deleted from or inserted into an existing subfield, it is convenient to represent the resulting subfield by a chain of two or more nodes. A facility to split subfields and merge parts of a subfield exists within IFORM, where other data needed for editing, e.g. current cursor position and first unused byte within the buffer are stored as well.

The 3RIP file structure is used to store records in EFORM on secondary storage. Characteristic of a 3RIP file is that it contains necessary information for processing, i.e. a directory, indicating the location of each record within the file, as well as some control information. As no special files are used for the directories, the number of separate files needed to run the 3RIP system is reduced.

Within a 3RIP file each record is named by a unique alphanumeric string. The directory is arranged as a B-tree (5), ordered on the name strings and as a consequence even large files can be searched and updated with fairly simple and efficient algorithms. When a record of a 3RIP file is modified by the editor, the new version of the record is added to the file, a new entry in the directory is created and the entry associated with the old version is marked for deletion. Thus new versions can be accessed directly and old versions are kept within the file and may also be retrieved. At suitable intervals the file is regenerated: current versions are written onto a new file and a new directory structure is established.

II. IMPLEMENTATION

The 3RIP record format and file structure could be implemented on any computer with good facilities for file, character, and bit string handling. In the following its implementation on a DEC-10 computer in the programming language BLISS(6) is described.

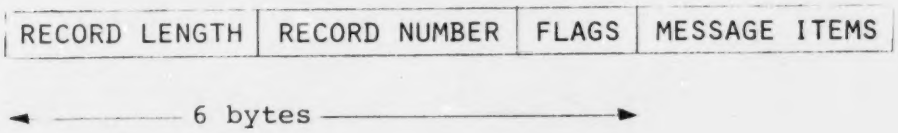
IIa. External record format (EFORM)

Each record consists of two parts, the *data part* and the *control part*.

The data part is divided into a variable number of data fields, each divided into any number of subfields of variable length. Characters are represented by 7-bit ASCII-code. All subfields are stored consecutively in one string. The string may be encoded by the 3RIP compression method, when the coded string is stored as a binary string using all 36 bits of the enclosed machine

words starting and ending at byte boundaries, however, with the last byte if necessary padded with zero bits.

The control part, defining the data contained in the data part consists of a *record head* and a *descriptor*. All fields within the control part are stored as 7-bit bytes. The record head has a fixed length part of 6 bytes possibly followed by a variable length part.



Record_length 2 bytes
A binary number indicating, in bytes, the length of the record, starting from the record number.

Record_number 3 2/7 bytes (23 bits)
A machine generated unique binary number.

Flags 5/7 bytes (5 bits)
Bits 1-4: Message flags.
This field indicates that the record has some kind of special contents which should be treated separately.

Bit 5: Coding status.
This field indicates whether or not the data field area of the record has been encoded.

Message items (need not exist)
This field corresponds to bits 1-4 of the Flags-field. If any of these flags is set to one this field contains information which is to be interpreted by one or more separate routines.

The descriptor consists of two parts, a data field indicator and length fields, each preceded by its size in bytes. The data field indicator size is stored in one byte and the extent of the length fields is given in another length field preceding them.

SIZE	DATA FIELD INDICATOR	LENGTH	LENGTH FIELDS

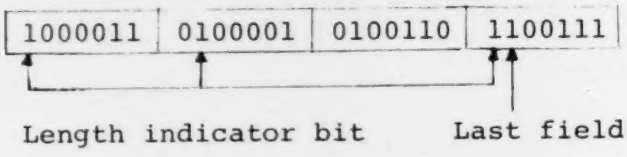
Data_field_indicator

The ordinal numbers of the bits set to one in the data field indicator coincide with the ordinal numbers of the data fields in the data part. The descriptor-entries and corresponding data fields within the data area appear in the same order.

Length_fields

To each bit set to one in the data field indicator there exists a length field entry which contains the length of the corresponding data field. When the data field contains several subfields the length of each sub-field appears in the length field entry. Length fields are of variable length containing binary numbers. The position of the leftmost bit set to one in such a field determines its length in bytes. This bit is called the length indicator bit. The last length field of an entry is so marked by having the bit following the length indicator bit set to one.

Ex. A length entry describing a data field divided into three subfields with lengths 3, 166 and 7 bytes respectively.



Records are stored in blocks of 512 36-bit words. If the size of a record exceeds 509 words it is divided into subrecords. In order to maintain the integrity of data fields when partitioning a long record the following rules apply:

Priority	Splitting point
1	Data field end
2	Data subfield end
3	Sentence end, where a sentence is considered ended by a period followed by either one or two spaces possibly followed by a capital letter.

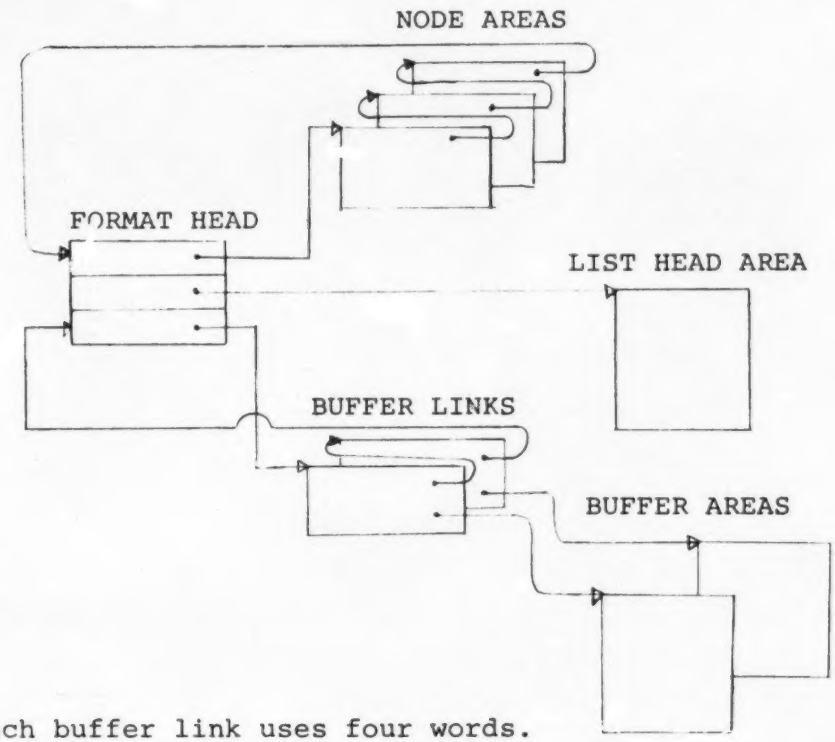
IIb. Internal format (IFORM)

The control part of the internal format consists of a list head area containing as many list heads as there are data fields defined for the particular file being processed. For each data field that is not empty the corresponding doubly-linked list contains nodes describing the subfields of the data field.

IFORM is implemented using a number of separate areas. The data part is stored in one or more *buffer* areas, linked together using separate buffer links. The *list heads* are stored consecutively in another area. All *nodes* are stored consecutively in a number of areas linked together using the first word of each area. Initially all nodes are linked together to form a list of free nodes. Information about the location of the areas of IFORM is contained in the *format head*, the address of which constitutes the address of IFORM. Moreover IFORM holds all information needed for editing, e.g. current cursor position, first free node to be

used and first free byte within each buffer.

Below is shown an IFORM representation using three node areas and two buffer areas.

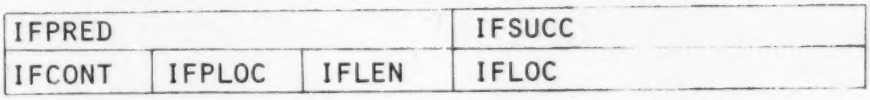


Each buffer link uses four words.

PRVB	NXTB
SIZEB	ADRB
POS	
	REST

Field	Size (bits)	Description
PRVB	18	Predecessor link
NXTB	18	Successor link
SIZEB	18	Buffer size
ADRB	18	Address of buffer
POS	36	First unused byte within the buffer
REST	18	Number of unused characters in the buffer

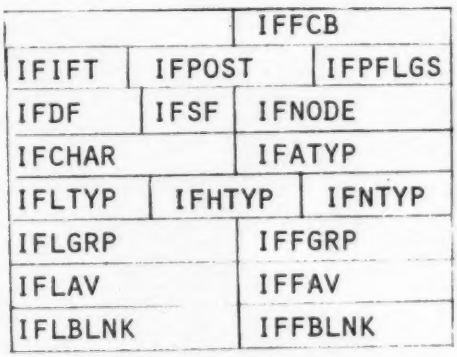
A node is represented as follows:



Field	Size (bits)	Description
IFPRED	18	Predecessor link
IFSUCC	18	Successor link
IFCONT	1	Continuation flag (used by the 3RIP editing system) If IFCONT is set to one then the current subfield is represented both by the current node and its successor in the list.
IFPLOC	3	Character position within the word pointed to by IFLOC
IFLEN	14	Size of field
IFLOC	18	Location of field (word address)

A list head is stored in one word, where the leftmost and rightmost 18 bits are used to store the address of the last and first node of the list respectively.

The format head occupies 8 words and is represented as follows:



Field	Size (bits)	Description
IFFCB	18	Address of file control block
IFIFT	8	Type of IFORM
IFPOST	23	Record number
IFPFLGS	5	Record head flags
		The following four fields are used to indicate current cursor position.
IFDF	9	Current data field number
IFSF	9	Current subfield
IFNODE	18	Current node
IFCHAR	18	Current character
IFATYP	18	Address of the list head area
IFLTYP	9	Lowest data field number present
IFHTYP	9	Highest data field number present
IFNTYP	9	Total number of data fields allowed for this file
IFLGRP	18	Address of last node area
IFFGRP	18	Address of first node area
IFLAV	18	Address of last free node
IFFAV	18	Address of first free node
IFLBLNK	18	Address of last buffer link
IFFBLNK	18	Address of first buffer link

When IFORM is used for the processing of an already existing record the sizes of the buffer, the list head and the node areas are chosen so that all of the information represented by EFORM can be accomodated. When a new record is created default values for the

size of the areas are used. However, while editing, when new character strings are inserted and new nodes are created, it may happen that either the buffer is filled or no more free nodes are available. In the first case a new buffer area is allocated and information about the area, e.g. size, location and first unused byte, is inserted into a buffer link, which is linked to the previous one. When more free nodes are needed a new node area is allocated.

IIc. File structure and file maintenance

On secondary storage the EFORM representation of the records are stored in blocks of 512 words. The structure of a block is shown below:

Unused bytes in a block are padded with binary zeroes.

Block head information

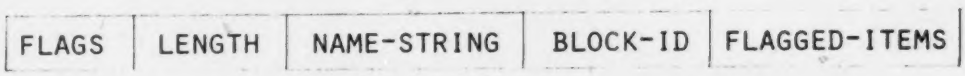
The block head consists of three words that might be followed by a variable length field.

File identifier	18 bits
Block number	18 "
Creation date	18 "
Modification date	18 "
Version	16 "
Number of free words in the block	10 "
Number of free bits in the block	6 "
Message flags	4 "
Message items (need not exist)	variable length

Each file consists of a *directory*, the *records proper*, and a *control record*, where parameters needed for the processing of the file are stored.

All records of a file are accessed through a directory consisting of record pointers, each giving the name and the location within the file of a particular record. The record pointers are sorted on the name strings and the directory is arranged as a B-tree structure. Each node in the tree is represented by a *node record* conforming to the 3RIP record format. The maximum size of a node record is 509 machine words, i.e. a record is contained in one block. The record pointers are stored as subfields of data field two. The node record representing the root is always stored in the first block of the file, and if the B-tree consists of more than one level the other node records are scattered within the file.

A *record pointer* is represented by



where all fields except the first are of variable length.

Flags	Interpretation of flags
Bit 1 - 2:	The record is to be processed by those routines that are specified within the FLAGGED-ITEM field.
3:	The record is protected against unauthorized access. The password or a password transformation is given in the FLAGGED-ITEM field.
4:	The record is long and has been split into subrecords. The number of blocks containing the subrecords is given in the FLAGGED-ITEM field.
5:	The record has been inverted and can be searched using the 3RIP search system (the flag is meaningful only for the primary file, BASE, of the search system).

6: Two different versions of a record are stored within the same block.

7: The record pointer is on the lowest level of the B-tree, i.e. the NAME-STRING and BLOCK-ID fields give the name and location of a record in the file. If the flag is zero the record pointer indicates the name of the last record pointer of a node record on a lower level of the B-tree and the location of this node record.

Length

The size of the next field.

Name-string

A record identifier, a character string for all files except the primary file where it is a binary number.

Block-id

The number of the block where the record is located.

Flagged-items

If either of the flags 1 or 2 are set the corresponding flagged-item(s) is (are) to be interpreted according to the rules specified in the control record. For the flagged-items associated with the flags 3 and 4 see interpretation of flags.

When a new record is entered the EFORM representation is appended to the file, and a record pointer is created and inserted into the directory. As all node records in the B-tree have a maximum length, a node is split into two when the insertion of a record pointer would imply that the contents of the node could not be contained

within 509 words. Thus, when the average size of a record pointer is small, the directory can normally be modified without the splitting of nodes.

When a new version of an existing record is entered or a record is deleted, the old or deleted versions are not removed but marked for deletion. This is indicated by storing the corresponding record pointers as subfields of data field three in the node record. The records marked for deletion are removed when the file is regenerated, i.e. when all current versions are written onto a new file and a new directory is established.

As an example of the use of the file structures consider a file of some four million records with an average size of 500 characters. If the 3RIP compression method is used the average size can be reduced to about 200 characters, and if an efficient technique for storing the records into blocks is used, some twelve records can be stored into a block of 2.560 characters. Then $3.3\ 10^5$ blocks would be used to store the records. A node record can contain some 400 record pointers, if the name string is represented as a binary number. This means that on the lowest level of the tree about 10^4 node records are needed and these records are reached using another 250 node records on upper levels. Thus the directory requires less than $0.2\ 10^5$ records and the file can be kept in $3.5\ 10^5$ blocks, i.e. $0.9\ 10^9$ bytes of disc space.

If the root of the B-tree is kept in core any record of a file of some 1.6 million records can be reached through two accesses and three accesses are needed for files containing up to 64 million records.

REFERENCES:

(1) Larsson, R. et al: *3RIP: An Interactive Search and Editing System for Large Textual Databases* (TRITA-LIB-4056, December 1976)

(2) Larsson, R. et al: *3RIP: File Design for the Search System* (TRITA-LIB-4042, March 1975)

(3) Hultgren, J. and Larsson, R.: *Pastime - A System for File Compression* (TRITA-LIB-4043, April 1975)

(4) Knuth, D.: *The Art of Computer Programming. Volume 1* (New York, Addison-Wesley, 1969)

(5) Knuth, D.: *The Art of Computer Programming. Volume 3* (New York, Addison-Wesley, 1973)

(6) Wulf, W.A. et al: *BLISS: A Language for Systems Programming* (*CACM*, Vol.14 No.12 December 1971)